An Interop Publication

CONNEXIONS®

The Interoperability Report

December 1994 Volume 8, No. 12

ConneXions — The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published monthly by Interop Company, a division of Softbank Exposition and Conference Company, 303 Vintage Park Drive, Foster City, California, 94404–1138, USA.
Phone: +1 (415) 578-6900
Fax: +1 (415) 525-0194
E-mail: connexions@interop.com

Subscription hotline: 1-800-575-5717
or +1-502-493-3217

ISSN 0894–5926

From the Editor

Our final issue of the year starts with an overview of *Pretty Good Privacy* (PGP). Largely the creation of a single person, Phil Zimmermann, PGP provides confidentiality and authentication services for electronic mail and file storage applications. While the Internet Engineering Task Force (IETF) has standardized *Privacy Enhanced Mail* (PEM), it is PGP and not PEM that has seen widespread deployment in the Internet and elsewhere. The article is by Bill Stallings, a frequent contributor to this journal and author of a just-published book on PGP.

Use of the Internet has traditionally been seen as being "free of charge" (or at least "fixed cost") from the perspective of its users. Of course, nothing is really free in this world, but the charges are often hidden from the end users and appear instead as per-month service provider costs, telco charges and so forth. However, sometimes it is desirable to share costs of an internet system by "fairly" charging each participating site based on the volume of traffic that they contribute. The University of Waikato in New Zealand has been operating a single Internet gateway to the U.S. on behalf of New Zealand universities and research institutions, charging users by volume to recover the costs. Our second article by Nevil Brownlee of The University of Auckland describes experiences of this charging model.

The DDN *Network Information Center* (NIC) has been operating an Internet directory service, known as WHOIS, for a number of years. WHOIS is used to provide a very limited lookup mechanism, serving information about a small number of registered Internet users. A working group of the IETF has been developing a new directory service, called WHOIS++. It provides a simple, distributed and extensible information lookup service based upon a small set of extensions to the original WHOIS information model. Chris Weider and Patrik Fältström of Bunyip Information Systems give an overview of WHOIS++.

As the year draws to an end, we wish all our readers happy holidays and remind you once again that we would love to hear from you. We truly value your comments and suggestions regarding anything you read in this journal. You can reach us most easily by sending e-mail to: **connexions@interop.com.** This address can be used for anything ranging from letters to the Editor to subscription questions. Other methods for reaching us are given on page 27. We are also constantly looking for new authors and topics for articles. Please contact us if you have any suggestions.

Pretty Good Privacy

by William Stallings

Introduction

Pretty Good Privacy (PGP) is a remarkable phenomenon. Largely the creation of a single person, Phil Zimmermann, PGP provides confidentiality and authentication services for electronic mail and file storage applications. In essence Zimmermann has done the following:

- Selected the best available cryptographic algorithms as building blocks.
- Integrated these algorithms into a general-purpose application independent of operating system or processor, based on a small set of easy-to-use commands.
- Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks such as Compuserve.
- Entered into an agreement with the company ViaCrypt to provide a fully compatible, low-cost commercial version of PGP.

Since its introduction in 1991, use of PGP has grown explosively and is now widespread. A number of reasons can be cited for this growth:

- PGP is available free worldwide in versions that run on a variety of platforms including DOS/Windows, UNIX, Macintosh, and many more. In addition, the commercial version satisfies users who want vendor support.
- It is based on algorithms that have survived extensive public review and are considered extremely secure. Specifically, the package includes RSA for public-key encryption, IDEA for conventional encryption, and MD5 for hash coding. (See [1] and [2] for a description of these three algorithms.)
- It has a wide range of applicability, from corporations that want to enforce a standardized scheme for file and message encryption to individuals who wish to communicate securely over the Internet and other networks.
- It wasn't developed by, nor is it controlled by, any governmental or standards organization. For those with an instinctive distrust of "the establishment," this makes PGP attractive.

Function	Algorithm Used	Description
Digital Signature	RSA, MD5	*A hash code of a message is created using MD5. This message digest is encrypted using RSA with the sender's secret key, and included with the message.*
Message encryption	IDEA, RSA	*A message is encrypted using IDEA with a one-time session key generated by the sender. The session key is encrypted using RSA with the recipient's public key, and included with the msg.*
Compression	ZIP	*A message may be compressed, for storage or transmission, using ZIP.*
E-mail compatibility	Base64 conversion	*To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using Base64.*
Segmentation	———	*To accommodate maximum message size limitations, PGP performs segmentation and reassembly.*

Table 1: Summary of PGP Services

The actual operation of PGP for sending and receiving messages (as opposed to key management functions), consists of five services: digital signature, message encryption, compression, e-mail compatibility, and segmentation (Table 1). Figure 1 traces the progress of a message through PGP from sender to receiver and illustrates all of these services, except segmentation.

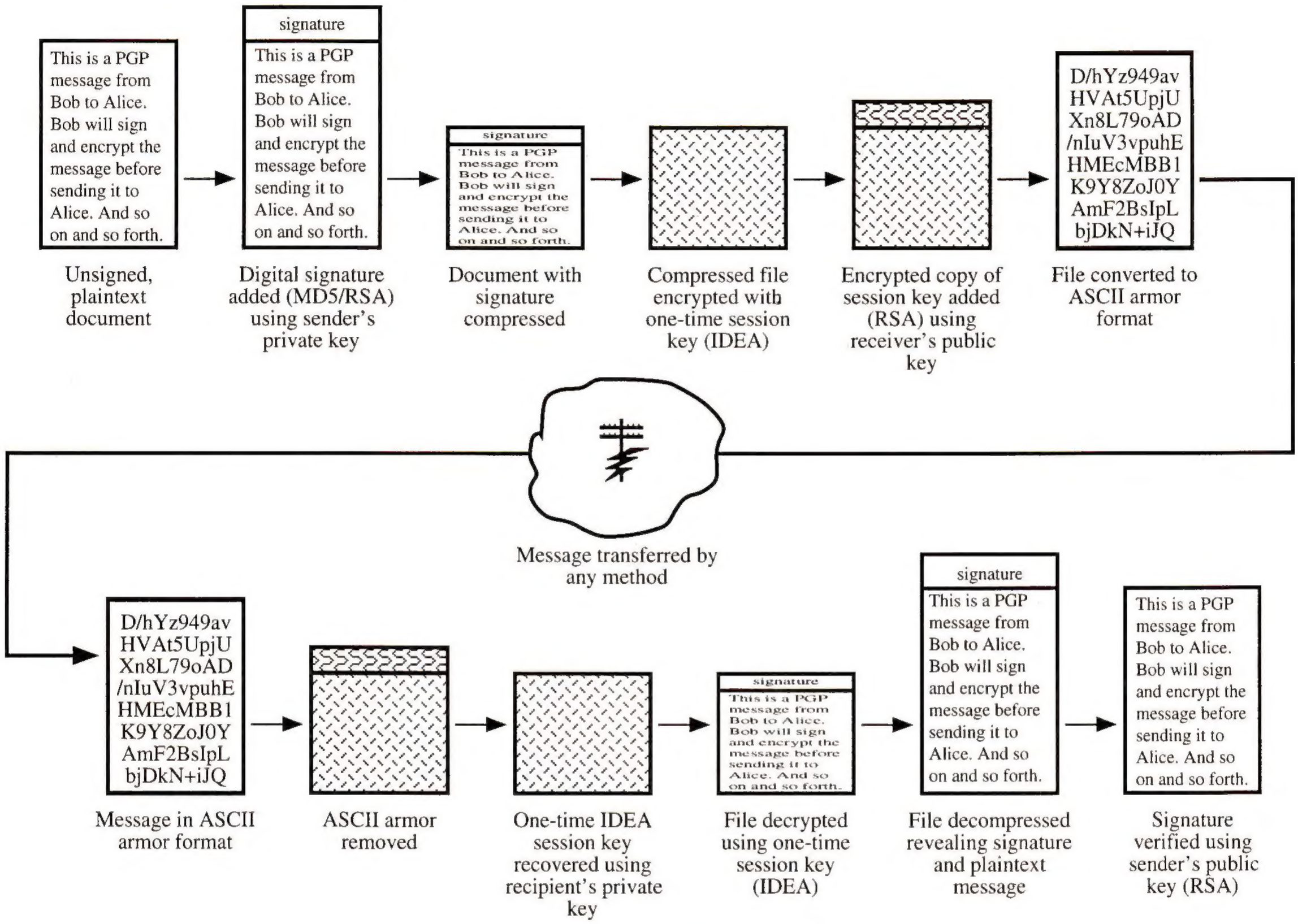

Figure 1: The Big Picture

Public Keys

Fundamental to the operation of PGP is the requirement that each user have a private key as well as a copy of the public key of each potential correspondent. PGP maintains a list of public keys that the user has obtained by one means or another. These keys are collected and stored on a public key ring. Each item on the ring actually includes several parts:

- The public key itself
- The User ID of the owner of this public key; typically the owner's name
- A Key ID, which is a unique identifier for this key
- Other information related to the trustworthiness of the key and its owner

The significance of the User ID and the key ID is that they provide two different ways of grabbing a key from the ring. PGP can retrieve a person's key from the public key ring given that person's name or given the Key ID. Both methods of retrieval are used, as shall be shown.

continued on next page

PGP *(continued)*

Private Keys

Each PGP must have a private key, and it is acceptable to have several private keys. So, the first thing a user should do after installing PGP is generate an RSA public/private key pair. When the two keys are generated, PGP places the public key on a data structure known as public key ring. As with other keys on that ring, it has a user ID (yours) and a key ID.

The other half of the pair, the private key, must be handled with more care. This key is to be stored on the user's private key ring. To secure the key, PGP doesn't simply store the private key on the private key ring. Instead, PGP asks the user for a passphrase, which is any sequence of characters made up by the user. It is called a passphrase rather than a password since it need not be a single word. An example of a passphrase is "T42andME4U." PGP then uses that passphrase to generate a 128-bit IDEA key (by taking the 128-bit MD5 message digest of the passphrase) and encrypts the private key using IDEA and the passphrase-based key. PGP stores the private key on the private key ring and discards the passphrase and the IDEA key.

The private key ring includes the following pieces of information for each private key:

- The private key, encrypted using the passphrase-based IDEA key
- The owner's User ID
- A copy of the matching public key.

To retrieve the private key the user must supply PGP with the passphrase. PGP reads the encrypted private key from the ring and decrypt it, using IDEA, with the key generated from the passphrase. Once this copy of the private key is used, it is immediately discarded. As a result of these precautions, even if someone steals a user's private key ring, it will do them no good without the passphrase.

Digital Signatures

Figures 2 and 3 indicate the operations involved in the sending of a message from Bob to Alice. The first step in the generation of a PGP message is the digital signature process, which is illustrated in the left-hand side of Figure 2. The sequence is:

- The sender creates a message.
- PGP uses MD5 to generate a 128-bit hash code of the message.
- The sender specifies the private key to be used for this operation and provides a passphrase, enabling PGP to decrypt the sender's private key.
- PGP encrypts the hash code with RSA using the sender's private key, and attaches the result to the message. The key ID of the corresponding sender's public key is attached to the signature.

The right-hand side of Figure 3 shows how a receiving PGP handles signatures:

- PGP takes the key ID attached to the signature and uses that to grab the correct public key from the public key ring.
- PGP uses RSA with the sender's public key to decrypt and recover the hash code.
- PGP generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.

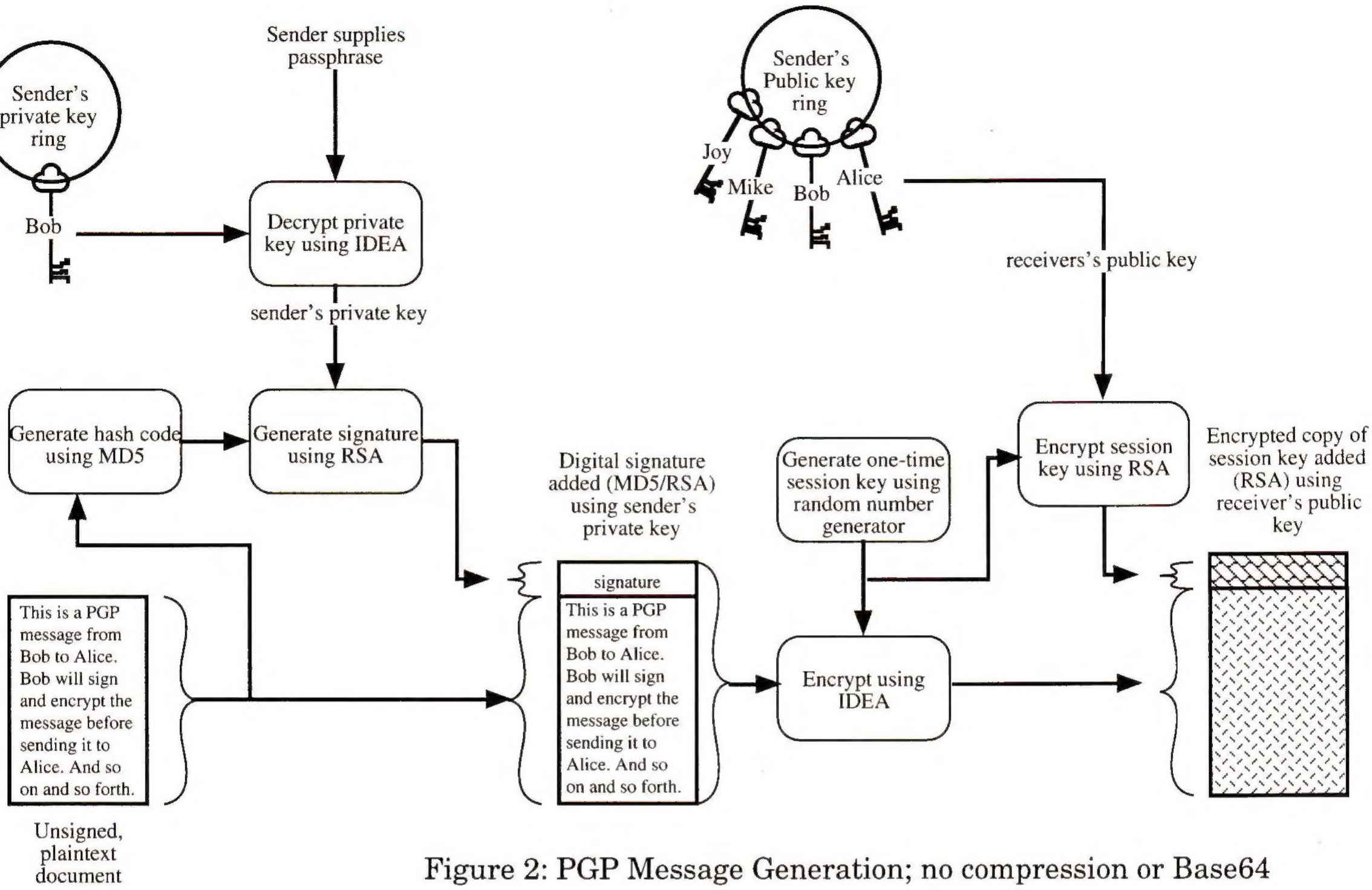

Figure 2: PGP Message Generation; no compression or Base64

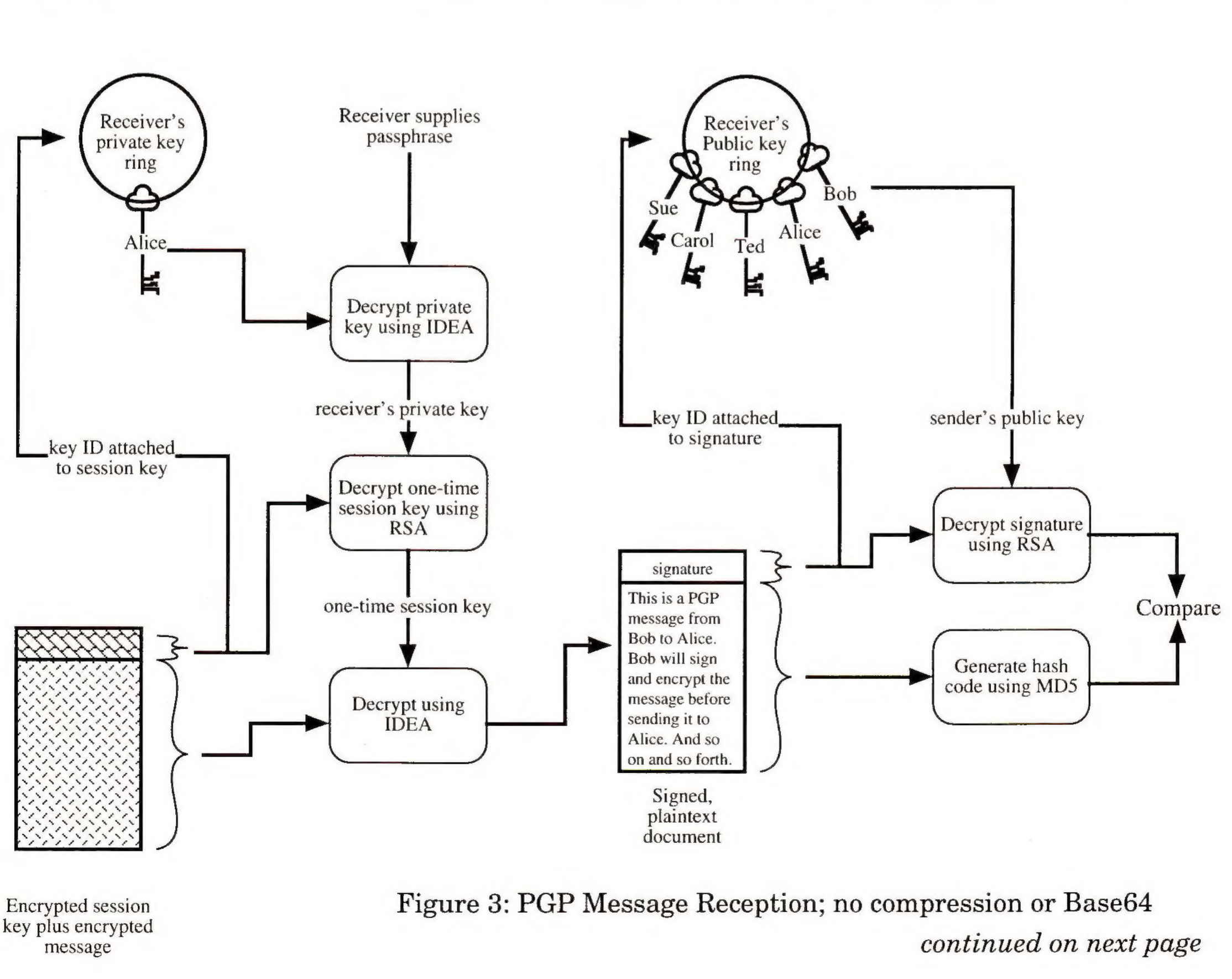

Figure 3: PGP Message Reception; no compression or Base64

continued on next page

PGP *(continued)*

Note that Alice's public key ring contains a copy of her own public key. There is normally no operational use of this key by Alice. However, from time to time, Alice will want to provide her public key to others and her public key ring is a handy place to store it.

The combination of MD5 and RSA provides an effective digital signature scheme. Because of the strength of RSA, the recipient is assured that only the possessor of the matching private key can generate the signature. Because of the strength of MD5, the recipient is assured that no one else could generate a new message that matches the hash code and, hence, the signature of the original message.

Compression

As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage.

PGP makes use of a compression package called *ZIP,* written by Jean-loup Gailly, Mark Adler, and Richard Wales. ZIP is a freeware package written in **C** that runs as a utility on UNIX and other systems. ZIP is functionally equivalent to PKZIP, a widely available shareware package developed by PKWARE, Inc. for MS-DOS systems. The ZIP algorithm is perhaps the most commonly used cross-platform compression technique; freeware and shareware versions are available for Macintosh and other systems as well as MS-DOS and UNIX systems.

In essence, the ZIP algorithm looks for repeating strings of characters in the input and replaces subsequent instances of such strings with compact codes. The destination system performs the same scanning function and is able to recognize the codes generated by the source and replaces these with the original text. The more redundancy there is in the input, the more effective the ZIP algorithm is in finding such sequences and the more heavily it can rely on codes. Typical text files compress to about one-half of their original length.

Another basic service provided by PGP is confidentiality, which is provided by encrypting messages to be transmitted or stored locally as files. In both cases the conventional encryption algorithm IDEA is used.

Message Encryption

As always, one must address the problem of key distribution. In PGP each conventional key is used only once; that is, a new key is generated as a pseudo-random 128-bit number for each message. Thus, although this key is referred to in the documentation as a session key, it is in reality a one-time key. Because it is to be used only once, the session key is bound to the message and transmitted with it. To protect the key, it is RSA-encrypted with the receiver's public key. The right-hand side of Figure 2 illustrates the sequence, which can be described as follows:

- PGP generates a pseudo-random 128-bit number to be used as a session key for this message only.
- PGP encrypts the message, using IDEA with the session key.
- PGP encrypts the session key with RSA, using the recipient's public key, and attaches the result to the message. The key ID of the recipient's public key is attached to the encrypted session key.

Note that Bob's public key ring differs from Alice's. A user's public key ring contains the public keys that this user has collected. Each user is responsible for collecting the public keys that he or she needs, so it is unlikely that two public key rings will have the same set of keys.

Turning to the receiving end (left-hand side of Figure 3):

- PGP takes the key ID attached to the message and uses that to grab the correct private key from the private key ring. Recall that a user may have more than one private key.
- The recipient provides a passphrase, enabling PGP to decrypt the recipient's private key.
- PGP uses RSA with the private key to decrypt and recover the session key.
- PGP decrypts the message using IDEA with the session key.

One can make several observations. First, to reduce encryption time the IDEA/RSA combination is used instead of just using RSA to directly encrypt the message; IDEA is substantially faster than RSA. Also, using RSA solves the session key distribution problem, because only the recipient is able to recover the session key that is bound to the message. Finally, the use of one-time conventional keys strengthens what is already a strong conventional encryption approach. Only a small amount of plaintext is encrypted with each key and there is no relationship among the keys.

As we have seen, both cryptographic services may be used for the same message. First, a signature is generated for the plaintext message and attached to the message. Then, after compression is used, the compressed plaintext message plus signature is encrypted using IDEA, and the session key is encrypted using RSA. In summary, when both services are used, the sender first signs the message with his/her own private key, then encrypts the message with a session key, and then encrypts the session key with the recipient's public key.

Base64 Conversion

When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private RSA key). If the confidentiality service is used, both the message and the signature (if present) are encrypted (with a one-time IDEA key). Thus, part or all of the resulting block consists of a stream of arbitrary eight-bit bytes. However, many electronic mail systems only permit the use of messages consisting of ASCII text. To accommodate this restriction, PGP provides the service of converting the raw data to a stream of printable ASCII characters. This conversion is done by mapping each group of three bytes of binary data into four ASCII characters; the format of the result is referred to as ASCII armor.

The use of ASCII armor expands a message by 33%. Fortunately, the session key and signature portions of the message are relatively compact and the plaintext message has been compressed. In fact, the compression should be more than enough to compensate for the ASCII armor expansion. We mentioned that ZIP produces a typical compression ratio of about 2 to 1. If we ignore the relatively small signature and key components, the typical overall effect of compression and expansion of a file of length X would be $1.33 \times 0.5 \times X = 0.665 \times X$. Thus, there is still an overall compression of about one-third.

One noteworthy aspect of the ASCII armor algorithm is that it blindly converts the input stream to ASCII armor format regardless of content, even if the input happens to be ASCII text. Thus, if a message is signed but not encrypted, and the conversion is applied to the entire block, the output will be unreadable to the casual observer, which provides a certain level of confidentiality.

continued on next page

PGP *(continued)*

As an option, PGP can be configured to convert to ASCII armor format only the signature portion of signed plaintext messages. This enables the recipient to read the message without using PGP, although PGP would still have to be used to verify the signature.

The Order of Operations in PGP

The order in which the five functions of digital signature, compression, message encryption, conversion to ASCII armor format, and segmentation are performed is critical. The signature is generated first, on the plaintext message, for two reasons:

- It is preferable to sign a plaintext message so that one can store only the plaintext message together with the signature for future verification. If one signed a compressed or encrypted file, then it would be necessary either to store that version of the message for later verification or to recompress and/or re-encrypt the message when verification is required.
- Even if one were willing to generate a recompressed message dynamically for verification, PGP's compression algorithm presents a difficulty. The algorithm isn't deterministic: Various implementations of the algorithm achieve different running speed versus compression ratio tradeoffs, and as a result produce different compressed forms. However, these different compression algorithms are interoperable, because any version of the algorithm can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all PGP implementations to exactly the same realization of the compression algorithm.

So the digital signature must come first, before compression or message encryption. The reason for performing message encryption after compression is to strengthen cryptographic security. The essence of a compression algorithm is to reduce redundancy in a block of message, which increases cryptographic security in two ways:

- Some forms of cryptanalysis rely on exploiting regularities in the plaintext to determine the key which has generated a given ciphertext. Compression tends to eliminate those regularities.
- When a brute-force attack is made, that is, when every possible key is tried, the attacker must examine each attempted decryption to determine success. If the plaintext is in compressed form, it may not be clear that a given decryption has yielded a valid plaintext. Of course, the attacker can perform a decompression on the result of each trial decryption, but this increases the cost of the brute-force attack.

Next, the ASCII armor conversion must be performed after digital signature, compression, and message encryption, so that the result is ready for transmittal over an e-mail facility. Finally, it is only after conversion to ASCII armor format that the final size of the message is attained, which is the appropriate time to apply segmentation.

Public Key Management

As can be seen from the discussion so far, PGP contains a clever, efficient, interlocking set of functions and formats to provide an effective confidentiality and authentication service. To complete the system, one final area needs to be addressed, that of public-key management.

Phil Zimmermann, in the PGP documentation, neatly captures the importance of this area:

> "This whole business of protecting public keys from tampering is the single most difficult problem in practical public key applications. It is the 'Achilles heel' of public key cryptography, and a lot of software complexity is tied up in solving this one problem."

In this area, PGP provides a structure for solving this problem, with several suggested options that may be used. This enables PGP to be used in a variety of formal and informal environments.

The essence of the problem is this: User A must build up a public-key file containing the public keys of other users in order to interoperate with them using PGP. Suppose that A's key file contains a public key attributed to B but that it is in actual fact owned by C. This could happen if, for example, A got the key from a bulletin board system (BBS) that was used by B to post the public key but that has been compromised by C. The result is that two threats now exist. First, C can send messages to A and forge B's signature, so that A will accept the message as coming from B. Second, any encrypted message from A to B can be read by C.

A number of approaches are possible within PGP for minimizing the risk that a user's public-key file contains false public keys. Suppose that A wishes to obtain a reliable public key for B. The following are some approaches that could be used:

- Physically get the key from B. B could store her public key on a floppy disk and hand it to A. A could then load the key into his system from the floppy disk. This is a very secure method but has obvious practical limitations.
- Verify a key by telephone. If A can recognize B on the phone, A could call B and ask her to dictate the key, in Base64 format, over the phone. As a more practical alternative, B could transmit her key in an e-mail message to A. A could have PGP generate a 128-bit MD5 digest of the key and display it in hexadecimal format; this is referred to as the "fingerprint" of the key. A could then call B and ask her to dictate the fingerprint over the phone. If the two fingerprints match, the key is verified.
- Obtain B's public key from a mutual trusted individual D. For this purpose, the introducer, D, creates a signed certificate. The certificate includes B's public key, the time of creation of the key, and a validity period for the key. D generates an MD5 digest of this certificate, encrypts it with her private key, and attaches the signature to the certificate. Because only D could have created the signature, no one else can create a false public key and pretend that it is signed by D. The signed certificate could be sent directly to A by B or D, or could be posted on a bulletin board or key server.
- Obtain B's public key from a trusted certifying authority. Again, a public key certificate is created and signed by the authority. A could then access the authority, providing a user name and receiving a signed certificate.
- Get B's key from a key server and verify the fingerprint, either directly with B or by monitoring public transmission of B. Many people make a practice of including their PGP fingerprint in postings to USENET newsgroups and other public forums.

continued on next page

PGP *(continued)*

For the third and fourth cases, A would have to already have a copy of the introducer's public key and trust that this key is valid. Ultimately, it is up to A to assign a level of trust to anyone who is to act as an introducer.

PGP Versions

Until the early part of 1994, life was fairly simple for the PGP user. The common denominator for all users was PGP version 2.3, available at a number of Internet ftp sites, through a number of commercial online services, such as Compuserve, and on numerous bulletin boards worldwide. Outside the U.S., there were no legal difficulties in using PGP. Inside the U.S., PGP 2.3 faced a legal problem. The package includes the RSA algorithm, for which there is a valid U.S. patent. Thus, any user of PGP 2.3 in the U.S. was vulnerable to being accused of patent infringement. One solution to this problem was the use of ViaCrypt PGP 2.4. ViaCrypt is a company that sells a supported version of PGP and has a sublicense from the RSA patent holder. For the U.S. user willing to pay for PGP, PGP 2.4 was available and fully interoperable with PGP 2.3.

But complications arose. In May of 1994, a group at MIT sanctioned by Phil Zimmermann issued a freeware version of PGP known as PGP 2.6. This version was released for noncommercial use with the agreement of the RSA patent holder and can therefore be used in the U.S. without risk of patent infringement. One problem with PGP 2.6 is that, since it was developed and deployed in the U.S., it cannot be legally exported without an export license.

However, PGP 2.6 quickly found its way outside the U.S., and there is nothing illegal about using the exported version; it was only illegal to export it. Another problem with 2.6 is that it doesn't fully interoperate with 2.3 and 2.4. PGP will decrypt messages and use keys generated by PGP 2.3 and 2.4. However, these earlier versions are unable to decrypt messages and use keys generated by PGP 2.6. MIT says that the reason for this incompatibility is to discourage use of the earlier software and mitigate the patent-caused problems that have hampered use of PGP within the U.S.

Several significant developments have occurred since the introduction of PGP 2.6. ViaCrypt has upgraded its products to Version 2.7, which is compatible with and interoperable with 2.6, 2.4 and 2.3.

For users outside the U.S. and Canada, several freeware versions have been developed One, developed in the U.K. is known as PGP 2.6ui. The "ui" stands for Unofficial International release because, unlike version 2.6, it hasn't been approved by Phil Zimmermann. Nevertheless, it is gaining in popularity. A more recent version is known as 2.6i, and has the approval of Phil Zimmermann.

The user thus has a number of choices. One likely question is: how safe are the various versions? That is, is there any sort of back door in any implementation that could be used by someone in the know to break the system? The developers of all these versions naturally assert that this isn't the case. For all freeware versions, the source code as well as the object code is available so that anyone can verify its integrity. For legal reasons, ViaCrypt doesn't provide source code, but the security of their version is endorsed by Phil Zimmermann. The personal opinion of the author is that all of these versions are safe, but you must make up your own mind.

What I can say is that PGP has attracted a large and devoted following, including many individuals and organizations such as the Electronic Frontier Foundation, among those who have a keen distrust of governmental and organizational attempts to invade privacy. The size of PGP's user base is a testimony to its security.

A final point: with the proliferation of implementations and versions, there is the danger of a lack of interoperability. To prevent that from happening, an effort is underway to formalize PGP's definition and to standardize PGP formats. Reference [3] is a first step in that direction.

Where to Get PGP

A frequently-updated list of sites for freeware PGP is available via anonymous ftp at `ftp.csn.net` in directory `/mpj` in file `getpgp.asc` (also found at `ftp.csn.net/mpj`). This lists a number of ftp sites in the U.S. and outside the U.S., a number of World Wide Web (WWW) facilities, and a few BBS sites. Another excellent source of pointers to PGP sites is through the Web; use the following URL: `http://www.mantis.co.uk/pgp/pgp.html`.

References

[1] Stallings, W., "Cryptographic Algorithms, Part I: Conventional Cryptography," *ConneXions,* Volume 8, No. 9, September 1994.

[2] Stallings, W., "Cryptographic Algorithms, Part II: Public-Key Encryption and Secure Hash Functions." *ConneXions,* Volume 8, No. 10, October 1994.

[3] Atkins, D. and Stallings, W., "PGP File Formats," RFC in preparation.

[4] Barlow, J. "A Plain Text on Crypto Policy," *Communications of the ACM,* November 1993.

[5] Schiller, J., "Issues in Internet Security," *ConneXions,* Volume 7, No. 9, September 1993.

[6] *ConneXions,* Volume 4, No. 8, August 1990, "Special Issue on Network Management and Network Security."

[7] Schiller, J., "Kerberos: Network Authentication, *ConneXions,* Volume 4, No. 1, January 1990.

[8] Kaliski, B., "An Overview of Public-Key Cryptography Standards," *ConneXions,* Volume 6, No. 5, May 1992.

[9] B. Clifford Neuman and Theodore Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications,* Volume 32, No. 9, September 1994.

WILLIAM STALLINGS is an independent consultant whose clients have included major corporations and government agencies in the United States and Europe. He is the author of over a dozen books on data communications and computers, including Network and Internetwork Security, from Prentice-Hall. He holds a PhD from M.I.T. in Computer Science and a B.S. from Notre Dame in electrical engineering. E-mail: `stallings@acm.org`

His PGP fingerprint is:

```
B1 4E 2A BD 96 08 8B A4
67 83 D1 09 FE 52 56 6C
```

New Zealand Experiences with Network Traffic Charging

by Nevil Brownlee, The University of Auckland

Abstract

Since 1990 the University of Waikato has operated a single Internet gateway to the U.S. on behalf of New Zealand universities and research institutions, charging users by volume to recover the costs. This has been very successful, allowing a steady growth in link speeds from 9,600 bps to 512 Kbps over four years.

The Internet backbone within New Zealand is provided by *Tuia,* an incorporated society of Research and Education users. Cost recovery for the backbone network within Tuia's membership is handled by several "management groups," with capacity-based settlements between the groups. This has proved to be a simple and effective scheme.

New Zealand's Internet Gateway

The New Zealand Universities are linked by the *Kawaihiko* network, one of Tuia's management groups. Initially Kawaihiko used a simple "equal shares" cost recovery scheme for internal traffic, which has worked well to date but is now somewhat limiting. Kawaihiko is moving to a new scheme, allowing each site to pay directly for its required traffic capacities. We believe this will encourage the development of new, bandwidth-intensive services.

New Zealand's link to the Internet began as a joint development project with NASA and formed part of the *Pacific Communications Programme,* PACCOM. The gateway itself was known as "the PACCOM gateway," but we have since renamed it "NZGate." NZGate is managed by Waikato University on behalf of Tuia, which is an incorporated society whose mission is to further the networking interests of the Research and Education community within New Zealand. "Tuia" is a Maori word which means "bound together."

NZGate began in April 1989 with a 9,600 bps analog cable link to Hawaii. Connectivity inside New Zealand was provided by PACNET (Telecom's public X.25 service) and Coloured Book software at each of the universities. PACNET access speeds were mostly 9,600 bps.

Goals

One slightly unusual aspect of NZGate was that although NASA provided generous support for the costs at the U.S. end of the link, no subsidy whatever was provided by the New Zealand government, which meant that all the link costs had to be recovered by charging the users. To get the project established, six of the universities agreed to pay 1/6 each of the startup and on-going costs. Shortly after this the universities agreed that volume-charging would be a better approach to sharing the costs. The underlying goals were:

- Measure traffic in both directions through NZGate for each participating site and charge for it by volume, i.e., for the number of Megabytes moved in and out each month
- Charge enough to cover actual costs, plus a percentage for development
- Use the resulting development funds to buy more capacity as demand grows

These goals provide an effective charging scheme for shared use of our single common link, but it is rather simple-minded. To make it usable we had to address several other matters, the first of which was predictability. Sites wanted to know well in advance how much they would have to pay for the NZGate service so that they could budget for it.

To provide predictability we adopted the notion of "committed traffic volume" per month. The charging scale has large steps, and the price per Megabyte decreases as the volume increases. Each site made an initial choice of their committed volume, and thus their monthly charge. The actual traffic was monitored month by month and reported back to all the sites. If a site's traffic falls into a different charging step for more than one month, that site's committed volume is changed to the actual rate. This allows a site to have a single unusual month, and it gives at least a month's warning of a change in the charge. It is simple to administer, since the committed volumes are changed automatically by the NZGate Management.

Another potential problem was that of paying for unsolicited incoming e-mail. As an example, a user at Auckland had a colleague in the U.S. who e-mailed him a 200 Kbyte file. The remote system kept on aborting after sending about 150 kilobytes, and did this every half hour over a three-day weekend. By the time we realised what was happening many tens of Megabytes had been received, which we had to pay for. We don't have a good answer to this—after all we have no control over systems outside New Zealand—but in more than four years operation it hasn't happened often enough to be worth worrying about.

Volume charging for NZGate was begun late in 1990 using metering software developed at Waikato University, running on a 286-based IBM PC. The metering has been progressively improved since then, and now runs on a Sun SPARCStation. The charging scheme has also been refined, with discounts to encourage off-peak use. A summary of the charge rates is given at the end of this article.

Growth

Traffic volumes have risen steadily, and the link capacity has been increased as follows:

March 1990	9,600 bps	analog cable to Hawaii
November 1990	14.4 Kbps	analog cable to Hawaii
September 1991	64 Kbps	satellite link to NASA Ames
February 1993	128 Kbps	satellite link to NASA Ames
March 1994	256 Kbps	digital cable to NASA Ames
July 1994	512 Kbps	digital cable to NASA Ames

Providers of intercontinental data circuits provide them as "half circuits," i.e., each end is separately billed. In our early stages PACCOM paid for the U.S. half circuit and NZGate paid for the New Zealand one. For some time now, however, NZGate has paid the charges for both ends. The New Zealand access circuit is provided by Telecom New Zealand; over the years we have used various intercontinental telecommunications providers.

Most of the above growth can be attributed to a steadily increasing user population, but there were also several events which caused sudden increases in traffic. The first of these was the setting up of the Kawaihiko network in April 1990, linking together all seven of the New Zealand universities. "Kawaihiko" is a Maori word, derived from "kawai" (a branching structure, like tree roots) and "hiko" (electricity). It began with Cisco routers at each university, linked with 9,600 bps leased lines which were later upgraded to 48 Kbps. It was an IP network (at last), which made it possible to use TCP/IP services such as FTP and greatly simplified access to USENET News.

continued on next page

Network Traffic Charging *(continued)*

Another event was the setting up of the Tuia network, which provided links between Kawaihiko, DSIRnet and MAFnet on an informal basis. This provided Internet access to all of New Zealand's government-funded research establishments, so increasing the number of sites using NZGate.

A third event occurred in July 1992, when the *Department of Scientific and Industrial Research* (DSIR) and *Ministry of Agriculture and Fisheries* (MAF) were restructured by the government and split into 11 *Crown Research Institutes* (CRIs). The Tuia Society was created, and a new network using a Frame Relay backbone was set up to link the Universities, CRIs, National Library and Ministry of Research, Science and Technology. The Frame Relay backbone provided significantly higher link capacities, which in turn boosted the total NZGate traffic.

During the formative phase of our shared use of the NZGate service, charging by volume has provided a cost-sharing mechanism which is visibly fair for each of the participating sites. The notion of "committed volumes" has smoothed out short-term usage transients, and thereby helped the budgeting process at each site.

The steady growth in transport volumes demonstrates the success of the service, and of its charging scheme. This is in stark contrast with the Chilean experience of charging. [1]

Management of the Tuia network

Before 1992 there were three Research and Education networks in New Zealand: DSIRnet (a leased-line network linking DSIR sites), MAFnet (a private X.25 network linking MAF sites) and Kawaihiko (a TCP/IP network linking the universities). These three national networks were interlinked so as to provide a single TCP/IP network known as Tuia. This network had no formal management structure, relying instead on occasional coordinating meetings of the three participants, but it nonetheless provided reliable, effective communication between the sites, and (via NZGate at Waikato University) access to the Internet.

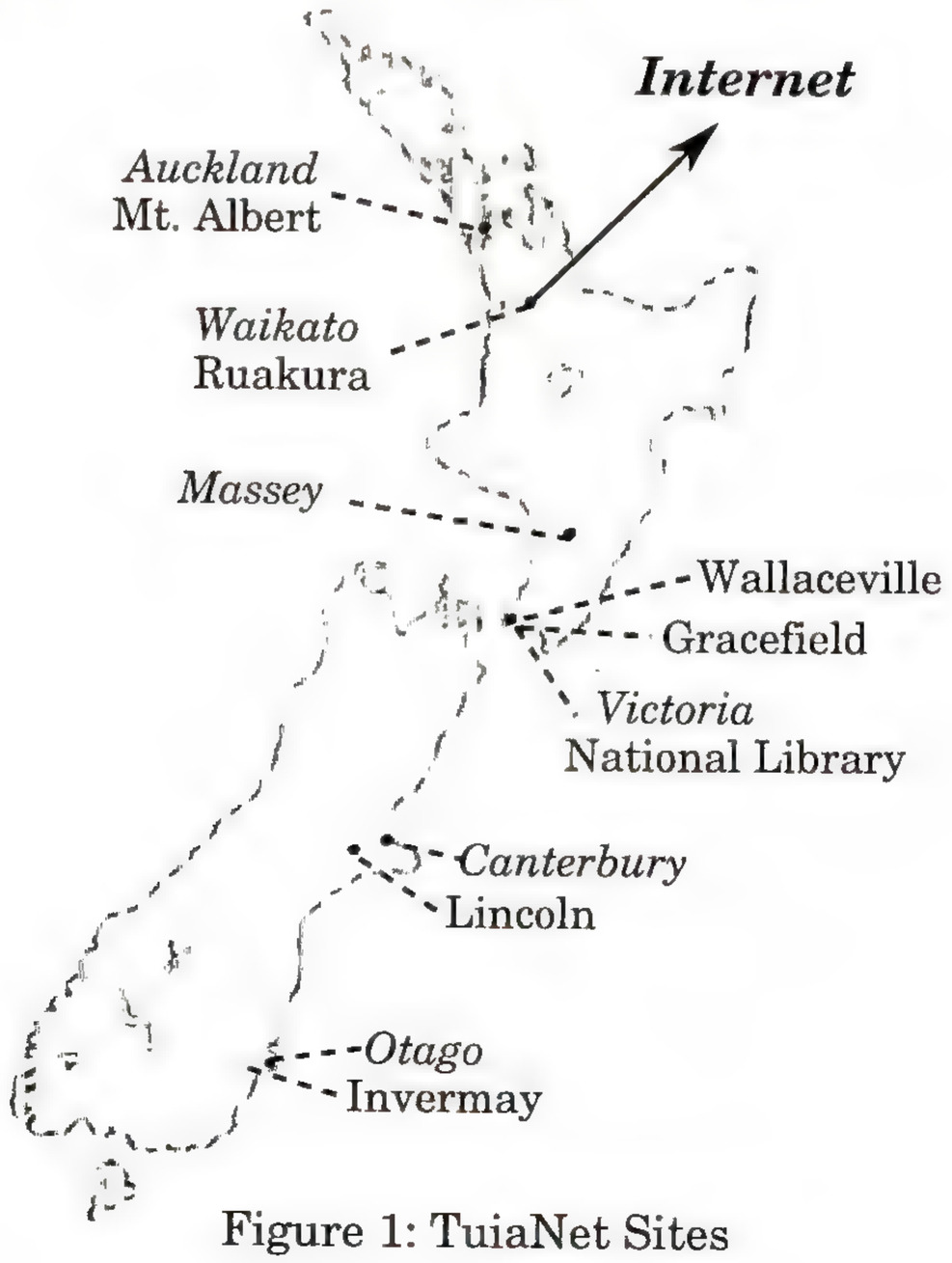

Figure 1: TuiaNet Sites

When the Tuia network backbone was set up in 1992 it linked 13 sites: five universities, six Crown Research Institutes (CRIs), the National Library of New Zealand and our Ministry of Research, Science and Technology (MoRST). The locations of the TuiaNet backbone sites are shown in Figure 1; the universities are labeled in *italics* and the CRIs in plain text.

Tuia is an incorporated society, providing its members (the CRIs, Universities, etc.) with a legal entity and a formal structure within which to set up and run a single backbone network, known as *TuiaNet*. Rather than set up a new management structure for TuiaNet we decided to continue with the old groupings of the sites, which became "management groups" within Tuia. Thus Kawaihiko is the Tuia management group which co-ordinates inter-university networking, a while *Industrial Research Limited* (IRL) and AgResearch co-ordinate groups of CRIs which correspond with the old DSIR and MAF. The remaining sites (National Library and MoRST) could have joined one of the three groups, but chose not to, which effectively makes them single-member groups.

TuiaNet uses a Frame Relay backbone provided by Netway Communications, a subsidiary of Telecom New Zealand. Netway's monthly charges for each site have two components, an access cost (determined by the line speed, 64 Kbps to 768 Kbps) and a set of *Committed Information Rate* (CIR) costs (reflecting the maximum continuous data rate for virtual circuits to other sites). CIR charges have been a flat rate of $8.75 per Kbps since we began using Frame Relay. Netway's monthly bills are paid by each of the management groups, and each group then recovers these costs from its own users as its management requires.

For sites which are entirely within a management group, e.g., the University of Auckland, the site pays its share of the costs to the group; as explained for Kawaihiko (see below).

Where a site provides services for more than one management group (e.g., Massey University provides backbone access for IRL and AgResearch), simple settlements have been reached. These usually have each group paying a fixed share of the access charge, and each group paying CIR costs to other sites within the group. A special case of this occurs when virtual circuits run between managements groups, e.g., between Waikato (Kawaihiko) and Gracefield (IRL). We have found that the traffic between members of a group is much greater than their traffic to other groups, so groups needing virtual circuits to other groups are happy to pay for the CIR at both ends.

The settlements are therefore capacity-based, in the sense that the amount each group pays is determined purely by the traffic capacity they agree to. They make no attempt to charge by volume, or for different network services; these possibilities are explored in the next section.

The simple arrangements for cost-sharing within Tuia have worked well. Having the management groups reduces the number of parties to settlements to a manageable number, allowing us to quickly reach agreement on the simple settlements. It seems unlikely that a flatter management structure, such as a single group which would then negotiate settlements between every pair of sites, would have been as effective.

continued on next page

Network Traffic Charging *(continued)*

Kawaihiko

Late in 1990 we began planning the Kawaihiko network, to link all the universities with their own IP network. Each site had a Cisco router, and the sites were interconnected with Digital Data Service (DDS) links. The topology was a central triangle joining Waikato, Victoria and Canterbury, with the others connected to it (as shown in diagram).

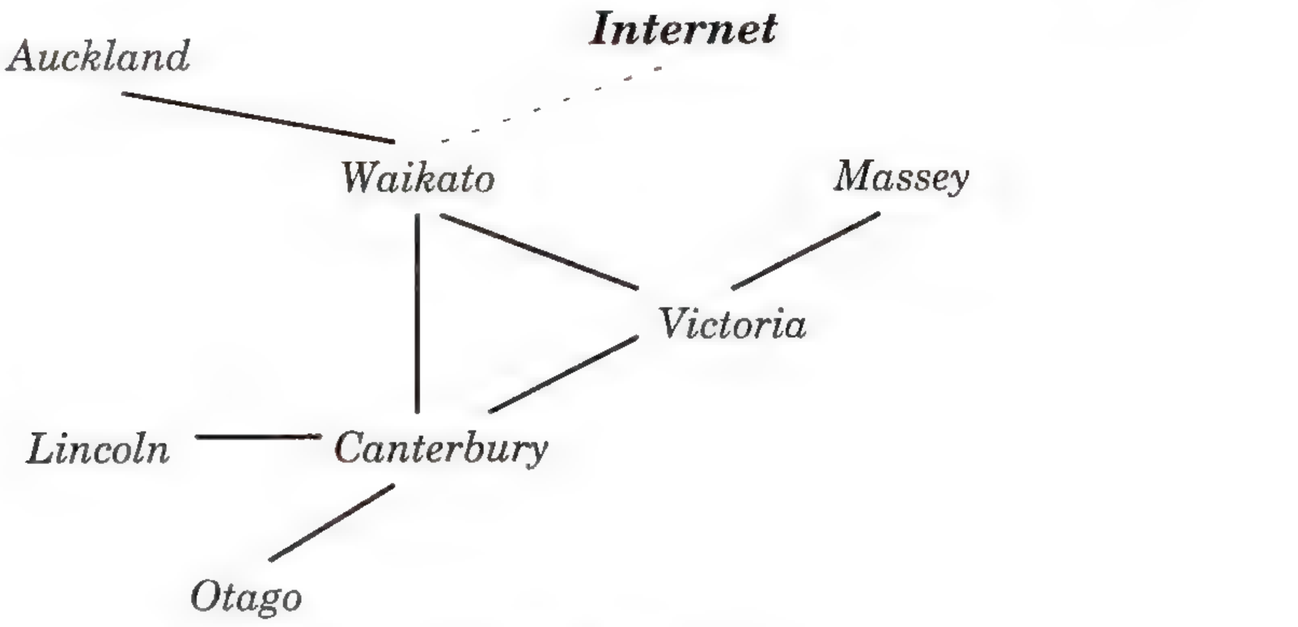

Figure 2: Kawaihiko Topology

The network was installed and running in April 1990, providing greatly improved communications between the universities. Its ongoing costs were Telecom's monthly access and transmission charges for the links. Following the success of volume charging for NZGate we resolved to implement volume charging to recover Kawaihiko's ongoing costs. It was not, however, possible to measure traffic volumes at that time, so instead we agreed—as an interim measure—to share the costs in fixed proportions, which were 1/13 for Lincoln and 2/13 for the other six sites.

This simple arrangement continued for about two years, during which time most of the links were upgraded to 48 Kbps. Waikato continued to measure traffic volumes through NZGate for each site. So as to provide metering of traffic between the Kawaihiko sites (rather than just the traffic on each of the Kawaihiko links), work was begun at Auckland in 1992 on an implementation of the Internet Accounting model. [2]

The Kawaihiko network was re-engineered in July 1992 when the Universities and Crown Research Institutes implemented TuiaNet, using Frame Relay as the dominant transport technology, but continuing to use DDS links where Frame Relay would have been too expensive. The effect of all this for Kawaihiko was to make it more obviously part of a larger network, with some of the Kawaihiko sites providing connectivity for some of the CRIs. Obviously a more elaborate cost-sharing scheme was called for.

A site requesting a frame relay connection must decide on its required access speed, and the CIR it wants to all the other sites. The access speed should be greater than the sum of the CIRs, and there are good reasons to have it much larger. One of these is that access speed determines the maximum burst rate, another is that it can be expensive to change access rates (especially if such a change involves changing the underlying technology, for example from DDS to fractional E1).

Since access rate had to be determined by each site separately, we agreed that each site would pay its own access cost. Access costs for sites providing common access for CRIs were divided using a set of percentages agreed locally at each site.

Initially we had to have a fully connected network, with at least 2 Kbps of CIR between every pair of sites. We therefore agreed to share the costs using our "1/13ths" formula, until we were able to measure the traffic volumes between sites.

The overall effect of these changes was that the Kawaihiko network was treated as being made up of links with access costs paid by individual sites, and transport costs shared between all sites—regardless of the type of link. Effectively, we have tried to visualise the New Zealand transport network as comprising local (and locally-funded) elements plus a single shared element whose costs were shared on the basis of traffic volumes, just as for the single international link element (NZGate).

Since 1992 work on the Internet Accounting project has continued, resulting in the public-domain release late in 1993 of an Accounting Meter, *NeTraMet,* and its Manager/Collector program, *NeMaC.* (*NeTraMet* is available via anonymous FTP from `ftp.auckland.ac.nz` in directory `pub/iawg/NeTraMet`).

There have been three further releases of *NeTraMet* in 1994, prompting considerable interest from other sites looking for a way of metering network traffic. Kawaihiko sites are now deploying *NeTraMet* meters on their gateway networks, and these will allow us to produce matrices showing traffic volumes between the sites, subdivided into a number of traffic types. If volume charging continues to be appropriate for all or part of the cost-sharing, these will provide the raw data for it. They will, in any case, provide data for other purposes, such as verifying that traffic flows don't exceed their agreed levels in the long term.

Initial work with the *NeTraMet* meters has classified traffic by site into traffic to and from "local," "other Kawaihiko sites (or networks connected to them)" and "world." Traffic types of particular interest so far include SMTP, NNTP, Telnet, FTP, Gopher, WWW and DNS.

Cost sharing within Kawaihiko

A charging scheme based on "fixed shares" has worked well so far, but as our traffic volumes grow and new services develop the need for a better algorithm becomes steadily more apparent. Charges based on amounts of data actually moved have been helpful throughout our "start-up" phase, during which institutions struggled to obtain funding for even a minimum level of service. This is essentially a cost-minimisation regime for the participating sites, and most appropriate for "queued" transfers whose transit time is not important. Not unnaturally, these have been dominant so far.

One of the requirements of a charging algorithm is that it should handle higher-layer services well. Some of these, such as *Domain Name Service* (DNS), and *Network Time Protocol* (NTP) are an essential part of the TCP/IP infrastructure. They must therefore be shared as "common good" elements of the network. Network News is somewhat similar in that all sites want access to it, and it is essential to minimise the number of times any news item is replicated. Overall it seems better to charge separately for transport (including "common good" services) and other "Value-Added" services. Service providers can be charged for their transport costs, and then recover them from their end users.

continued on next page

NEVIL BROWNLEE holds an M.Sc. and a Ph.D in Atmospheric Physics from the University of Auckland. Since 1975 he has been Deputy Director of the University of Auckland's Computer Centre, where he is responsible for a campus network with about 2,500 connected hosts. He is manager of Kawaihiko and chairman of Tuia's Technical Working Group. He has been active in the IETF since 1992, and is currently co-chair of the Operational Statistics and Internet Accounting Working Groups. His hobbies include backstage Theatre (especially Lighting Design), photography and music. E-mail:
n.brownlee@auckland.ac.nz

Network Traffic Charging *(continued)*

As an example of this, consider a site wishing to offer a *World-Wide Web* (WWW) cache as a chargeable service. Transport costs for traffic to and from the cache could be measured and paid for by the cache provider, who would set charges for the service so as to cover the transport charges both within New Zealand (to the customers) and to NZGate (source of the cached Web pages).

Another requirement is the emerging need to provide higher-bandwidth pipes within Kawaihiko which are site-specific, e.g., pairs of sites such as Waikato and Victoria would like 128 Kbps capacity between them for experiments with packet video. These "dedicated capacities" should clearly be paid for by the sites requiring them, and not be a shared cost for all of Kawaihiko. Charging directly for capacity provides predictable costs for sites, and allows for special link needs. Specified capacities appear to be the most effective way to provide for the continued growth of Kawaihiko, and are the basis of our proposed new charging algorithm, as follows.

Each site will nominate the required capacities for its links to every other site. From the lists of link capacities we will produce a full matrix—not necessarily symmetric—of actual link capacities (CIRs or line speeds) which will be sufficient to meet the stated requirements. In most cases these will be the CIRs we need to meet our current traffic levels plus any anticipated needs for the next year or so.

Each site will pay its own access costs, whether they are connected via Frame Relay or DDS lines. Sites will continue to make their own arrangements with distal (i.e., locally connected) sites. For these aspects, cost recovery simply continues our existing practice.

Transport costs will be paid by the sites. For each link or virtual circuit the costs will be shared between the two sites involved. This cost sharing may be equal, which amounts to having each site paying all its CIR costs directly. On the other hand it may be unequal, e.g., if an Auckland distal (locally connected) site wants 128 Kbps of CIR to Waikato, Auckland could pay Access and 128 Kbps CIR costs for both ends of the virtual circuit, then recover these from the distal site.

Conclusion

Over the last four years New Zealand has had a very effective Internet Gateway, the speed of which has steadily increased to meet users' traffic demands. Because we have charged users by volume for their traffic, users have always perceived that their payments were closely related to the benefits they derived. This perception was enhanced by our ability to develop our Internet gateway so as to handle the increasing traffic volumes.

The overheads of charging are significant. Operation and development of the NZGate traffic monitoring and usage billing software occupies about one half-time person. Development of *NeTraMet* is an ongoing research activity, taking about one-fifth of my time. Management of Kawaihiko takes one person about half a day each week. The benefits provided by charging are, however, well worth their cost to us.

Within New Zealand we have used "fixed share" algorithms to recover costs, which has worked well within our simple management structures. Kawaihiko, the universities management group, have invested a great deal of effort in developing traffic accounting and cost sharing methods, and are currently implementing a "specified capacity" algorithm for sharing their traffic costs. We believe this will more accurately reflect our users' expectations, allowing us to handle the newer, more bandwidth-intensive services they demand.

Acknowledgments

The author records his appreciation to the New Zealand Vice Chancellors' Committee for their support of networking within our universities, to NASA for their enthusiastic support of our international Internet gateway in its early stages, and to the network support staff at all the New Zealand sites, especially to John Houlker. It was John who made our first contact with NASA and PACCOM, implemented the gateway, and has managed it so effectively ever since.

NZGate charging rates

The NZGate International Internet Charging rates as of April 8, 1994 are given below. More information can be obtained from `nic@wakato.ac.nz`.

Uncommitted rate: $6.00 per MByte

Committed rates:

Commitment (in Mbyte)	*$/Mbyte*	*Fee (in $)*
100	4.00	400
200	3.80	760
300	3.70	1,110
400	3.60	1,440
500	3.50	1,750
600	3.40	2,040
700	3.30	2,310
800	3.20	2,560
900	3.10	2,790
1,000	3.00	3,000
1,250–1,000	2.90	3,625
1,500–1,250	2.80	4,200
2,000–1,500	2.60	5,200
2,500–2,000	2.50	6,250
3,000–2,500	2.50	7,500
3,500–3,000	2.50	8,750
4,000–3,500	2.50	10,000
4,500–4,000	2.40	10,800
5,000–4,500	2.40	12,500
5,500–5,000	2.40	13,200
6,000–5,500	2.30	13,800
6,500–6,000	2.30	15,000
7,000–6,500	2.30	16,000

- *Committed rate buffer period:* If the committed rate is exceeded for a single month, the charge step remains unchanged (and similarly if it is not reached for a single month).
- *Low priority discounts:* Low priority traffic (currently FTP and MAIL) is discounted by 30%. Normal priority traffic is discounted by 15%. The full rate is charged for high priority traffic (currently TELNET and FTP commands) These are applied before the committed rate is determined.
- *Off peak discount:* An off peak discount of 80% applies to all traffic between 8pm and 9am. This is applied after the committed rate is determined.

References

[1] Ricardo Baeza-Yates, Jose M Piquer, Patrico V. Poblete, "The Chilean Internet Connection or I Never Promised You a Rose Garden," Proceedings of INET '93.

[2] C. Mills, D. Hirsh, G.R. Ruth, "Internet accounting: Background," RFC 1272, November 1991.

[3] D. Hirsh, "Internet Resource Accounting," *ConneXions,* Volume 5, No. 7, July 1991.

The WHOIS++ Directory Service

**by Chris Weider and Patrik Fältström,
Bunyip Information Systems, Inc.**

Directory Services in the 1990s

In many ways, the Internet infrastructure for locating information and people in the mid-'90s looks a lot like the networking infrastructure did in the mid-'80s. There are lots of special purpose directory services, such as *archie,* which indexes information about files available via FTP. Until recently, however, there was only one protocol which was designed to provide a global directory service for people and resources, the X.500 protocol. A number of technical and other problems with the X.500 protocol and its implementations inspired the creation of the *WHOIS++ Directory Service* protocol in mid-1992.

What is WHOIS++ ?

The WHOIS++ protocol provides a globally distributed, easily searchable directory service for people and resources. It was originally developed as a set of extensions to the WHOIS protocol, which is still in use on the Internet, but since its original development in mid-1992 it has evolved into a completely separate protocol.

The architecture of WHOIS++ contains two components; a base level server, and an indexing service. The base level server is described in [1], the indexing service in [2].

WHOIS and WHOIS++

The current NIC WHOIS service [3] is used to provide a very limited directory service, serving information about a small number of Internet users registered with the DDN NIC. The basic WHOIS information model represents each individual record as a Rolodex-like collection of text. Each record has a unique identifier (or handle), but otherwise is assumed to have little structure. The current service allows users to issue searches for individual strings within individual records, as well as searches for individual record handles using a very simple query–response protocol.

WHOIS++ provides, a simple, distributed and extensible information lookup service based upon a small set of extensions to the original WHOIS information model. These extensions allow the new service to address the community's needs for a simple directory service, yet the extensible architecture is already allowing it to find application in a number of other information service areas.

Added features include an extension to the trivial WHOIS data model and query protocol and a companion extensible, distributed indexing service. A number of other options have also been added, like Boolean operators, more powerful search constraints and search methods.

The basic query syntax has been enhanced to make both the client and the server part of the dialogue more stringent and parsable. An optional authentication mechanism for protecting all or parts of the associated WHOIS++ information database from unauthorized access is also included in the protocol.

The basic architecture of WHOIS++ allows distributed maintenance of the directory contents and the use of the WHOIS++ indexing service for locating additional WHOIS++ servers.

The Data Model

WHOIS++ entries are structured around a series of standardized information templates, such as those described by [4]. Such templates consist of ordered sets of data elements (or attribute-value pairs) and a number of groups in the *Internet Engineering Task Force* (IETF) are now working on standardizing their format and content for various applications.

Current templates include *User, Network,* and *Organization,* and others are under development for other types of resource location applications. These templates are designed to allow rapid application to existing directory technologies, such as flat files and specialized databases. The creation and use of customized templates should also be possible with little effort, although their use should be discouraged where appropriate standardized templates exist, as template management issues can cause problems in a distributed environment.

The Query Syntax

WHOIS++ users may constrain searches to desired attributes or template types, in addition to the grandfathered WHOIS commands for specifying handles or simple strings. There are also a number of search term modifiers to allow individual search terms to be matched in different ways, such as "leading substring," "case sensitive." etc. A sample query is included below.

```
Name=Weider, search=fuzzy; Address=Michigan
```

In addition to search queries, WHOIS++ servers can respond to requests for blank templates of a given type, to allow the user to see what attributes are associated with a given template.

WHOIS++ servers are also required to contain a "DESCRIBE" template, which contains information about the server in free text form, and which can be retrieved by the client. This allows easy identification of a given server, and may help the user select which servers they wish to query.

The Index Service

The indexing service is designed to solve a number of problems associated with searching in other distributed directory services and distributed databases.

One of the primary assumptions made by recent implementations of distributed directory services (particularly in the X.500 protocol) is that every entry resides in some location in a hierarchical name space. While this arrangement is ideal for reading the entry once one knows its location, it is not as good when one is searching for the location in the namespace of those entries which meet some set of criteria. If the only criteria we know about a desired entry are items which do not appear in the namespace, we are forced to do a global query.

Whenever we issue a global query (at the root of the namespace), or a query at the top of a given subtree in the namespace, that query is replicated to all subtrees of the starting point. The replication of the query to all subtrees is not necessarily a problem; queries are cheap. However, every server to which the query has been replicated must process that query, even if it has no entries which match the specified criteria.

This part of the global query processing is quite expensive. A poorly designed namespace or a thin namespace can cause the vast majority of queries to be replicated globally, but a very broad namespace can cause its own navigation problems. WHOIS++ alleviates this problem by creating an "index mesh" which can efficiently and effectively route any query to servers which will be able to resolve it.

Index meshes and forward knowledge

We'll hold off for a moment on describing the actual architecture used in our solution to these problems and concentrate on a high level description of what solutions are provided by our conceptual approach.

continued on next page

The WHOIS++ Directory Service *(continued)*

To begin with, although every entry in WHOIS++ does indeed have a unique identifier (resides in a specific location in the namespace) the navigational algorithms to reach a specific entry do not necessarily depend on the identifier the entry has been assigned.

The Index Service gets around the namespace and hierarchy problems by creating a directory mesh on top of the entries. Each layer of the mesh has a set of "forward knowledge" which indicates the contents of the various servers at the next lower layer of the mesh. Thus when a query is received by a server in a given layer of the mesh, it can prune the search tree and hand the query off to only those lower level servers which have indicated that they might be able to answer it. Thus search becomes feasible at all levels of the mesh.

In the current version of this architecture, we have chosen a certain set of information to hand up the mesh as forward knowledge. This may or may not be exactly the set of information required to construct a truly searchable directory, but the protocol itself doesn't restrict the types of information which can be handed around.

In addition, the protocols designed to maintain the forward knowledge will also work perfectly well to provide replication of servers for redundancy and robustness. In this case, the forward knowledge handed around by the protocols is the entire database of entries held by the replicated server.

Another benefit provided by the mesh of index servers is that since the entry identification scheme has been decoupled from the navigation service, multiple hierarchies can be built and easily maintained on top of the existing data. Also, the user does not need to know in advance where in the mesh the entry is contained.

Since the index servers can pick and choose between information proffered by a given server; and because we have an architecture that allows for automatic polling of data, special purpose directories (for Yellow Pages applications, for example) become easy to construct and to maintain.

To participate in the Index Service, a server must only know about the WHOIS++ protocol described in [1]. This implies that any underlying database can participate, as long as one is able to communicate with some front-end that can generate a "centroid" or some other type of forward knowledge. A database might not be able to store other centroids, i.e., it can be a "leaf-server."

Some others might not be able to store anything other than centroids ("index-servers") and some will be able to do both. The WHOIS++ protocol is the only restriction we have put on participating databases. Also, the protocol itself has a very small amount of core functionality that must be implemented. Other parts of the protocol (for example complex searches with regular expressions) is an extension that one might implement if one wants. Allowing non-WHOIS++ servers to participate in the Index Service is being discussed in the IETF.

Centroids

As we mentioned above, there are many different types of indexing information which will be useful for pruning the indexing mesh. We are designing the indexing protocol so that all the different types of indexing information can coexist on a single index server. The first one which has been standardized is the "centroid."

The centroid of a server is comprised of a list of the templates and attributes used by that server, and a word list for each attribute. The word list for a given attribute contains one occurrence of every word which appears at least once in that attribute in some record in that server's data, and nothing else.

For example, if a WHOIS++ server contains exactly three records, as follows:

```
Record 1
Template:User
First Name: John
Last Name: Smith
Favourite Drink: Labatt Beer

Record 2
Template: User
First Name: Joe
Last Name: Smith
Favourite Drink: Molson Beer

Record 3
Template: Domain
Domain Name: foo.edu
Contact Name: Mike Foobar
```

the centroid for this server would be:

```
Template:          User
First Name:        Joe
                   John
Last Name:         Smith
Favourite Drink:   Beer
                   Labatt
                   Molson
Template:          Domain
Domain Name:       foo.edu
Contact Name:      Mike
                   Foobar
```

It is this information which is handed up the tree to provide forward knowledge. As we mention above, this may not turn out to be the ideal solution for forward knowledge, and we suspect that there may be a number of different sets of forward knowledge used in the Index Service. However, the directory architecture is in a very real sense independent of what types of forward knowledge are handed around, and it is entirely possible to build a unified directory which uses many types of forward knowledge.

Current implementations

There are several freely available implementations of the server code, however, the Internet Drafts describing the WHOIS++ protocol were issued as Draft Standard only in November 1994, thus they may need some minor tweaking to comply with the current version of the standard. Funding has recently been granted to provide reference implementations of WHOIS++ servers and clients, and to deploy a large-scale test bed.

The publicly available implementations should be widely available in First Quarter 1995. In addition, the current implementations have also been used to provide WWW gateways to WHOIS++, and WHOIS++ is being tested for use in other information infrastructure applications.

continued on next page

The WHOIS++ Directory Service *(continued)*

For more information on the current status and location of WHOIS++ implementations, please contact the authors. Potential implementors are also encouraged to retrieve the standard and implement it; it has been deliberately kept simple for experimental implementations.

The future of WHOIS++

WHOIS++ is under very active development, and some modest deployment. Full standardization in the IETF should take place in 1995. The freely available reference implementations should help widen deployment, and commercial versions of WHOIS++ will probably be available in Fourth Quarter 1995. WHOIS++ will also be a major participant in the *Shared Internet White Pages,* which is currently under development at the IETF. WHOIS++ has the potential to address a number of the Internet Directory problems which have surfaced in the last several years, and it is also being investigated for use in resource location and discovery applications on the Internet.

Conclusions

It is expected that the minimalist approach we have taken will find application where the high cost of configuring and operating traditional White Pages services can not currently be justified. Also note that the new architecture makes no assumptions about the search and retrieval mechanisms used within individual servers. Operators are free to use dedicated database formats, fast indexing software or even provide gateways to other directory services to store and retrieve information, if desired.

The WHOIS++ server simply functions as a known front end, offering a simple data model and communicating through a well known port and query protocol. The format of both queries and replies has been structured to allow the use of client software for generating searches and displaying the results. At the same time, some effort has been made to keep responses at least to some degree readable by humans, to ensure low entry cost and to ease debugging. The actual implementation details of of an individual WHOIS++ search engine are left to the imagination of the implementor and it is hoped that the simple, extensible approach taken will encourage experimentation and the development of improved search engines.

References

[1] Deutsch, Peter, et. al., "Architecture of the WHOIS++ service," Internet Draft, July 1994.

[2] Weider, Chris et. al., "Architecture of the WHOIS++ Index Service," Internet Draft, July 1994.

[3] Harrenstien K., Stahl M., Feinler E., "NICNAME/WHOIS," RFC 954, October 1985

[4] Emtage A., & Deutsch P., "Publishing using the Internet Anonymous FTP Archives Templates," Internet Draft, November 1994.

[5] Benford, Steve, "Components of OSI: X.500 Directory Services," *ConneXions,* Volume 3, No. 6, June 1989.

[6] Deutsch, P. & Emtage, A., "The *archie* System: An Internet Electronic Directory Service," *ConneXions,* Vol. 6, No. 2, Feb. 1992.

PATRIK FÄLTSTRÖM holds a MSc in Mathematics from the University of Stockholm, 1991. He is on the board of the Swedish Network Users Society (SNUS), and is test leader for the E-Mail tests held at the annual SNUS "Interoperability" event. He is also a member of IETF and as such he has been working with standardizing the MIME specification and distributed directory services. He is advising the Swedish Government in writing a technical specification for a Swedish standard of addressing electronic mail (X.400 and SMTP). Currently he is working as a Staff Researcher at Bunyip Information Systems, Inc. on the development of the WHOIS++ service. He can be reached at `paf@bunyip.com`

CHRIS WEIDER holds a BS in Mathematics and a BS in Computer Science from the University of Missouri, and an MA in Mathematics from the University of Michigan. He is one of the primary developers of the WHOIS++ protocol, and is also well-known for his many papers on Internet information services architecture. He has chaired several IETF working groups and still chairs the Integration of Internet Information Resources group. His research interests include distributed indexing, Internet search technology, natural language processing, and artificial intelligence. He is currently Senior Manager of New Business Development for Bunyip Information Systems, Inc. and is also responsible for WHOIS++ development. E-mail: `clw@bunyip.com`

Call for Papers

The international data communications research journal *Computer Communications* announces a special issue on Algorithms for ATM Networks. The Guest Editor is Dr. Bala Rajagopalan of AT&T Bell Laboratories.

ATM has been widely acknowledged as the technology for broadband ISDN. Many local and wide-area ATM networks are in use and more are on the way to deployment. While progress has been made in standardizing ATM network access, much work is still underway on resolving many key networking issues.

Topics

This special issue of *Computer Communications* aims to present and document current research and experience in the design, analysis and implementation of ATM network algorithms. The focus will be on practical algorithms for routing, traffic management, internetworking and allied functions. Relevant topics include:

- Virtual circuit routing and admission control
- Multicasting
- Internetworking
- Resource allocation
- Congestion control
- Policing
- Dynamic bandwidth sharing
- Queueing disciplines
- Performance models
- Implementation experience

Important dates

Submissions due:	February 28, 1995
Author notification:	April 30, 1995
Final Manuscripts:	June 30, 1995
Publishing date:	Autumn 1995

Author information

Submissions made to the special issue should not have appeared in, or been submitted to other archival publications. All papers will be subjected to the journal's usual refereeing process. Papers developed from earlier conference and workshop presentations are welcome. Prospective authors should send six copies of their manuscript (in English) to the guest editor:

Dr. Bala Rajagopalan
AT&T Bell Laboratories, Room 1F-401A
101 Crawfords Corner Road
Holmdel, NJ 07733
USA
Tel.: +1 908 949 8017
Fax: +1 908 949 1726
E-mail: `braja@qsun.att.com`

Authors are advised to consult the journal's "Notes for Authors Submitting on Disk," published in the journal or available from the General Editor (PO Box 31, Market Harborough, Leics LE16 9RQ, UK) or from the US Editor (Raj Yavatkar, Dept. of Computer Science, University of Kentucky, 40506–0046, USA, `raj@dcs.uky.edu`) before submitting their papers. Publication guidelines are also available in the World-wide Web, at `http://www.elsevier.nl/`

Call for Papers

The *ACM SIGCOMM '95* Conference will be held in Cambridge, Massachusetts, USA August 30 to September 1, 1995. (Tutorials and Workshop, August 28 and 29). SIGCOMM is an international forum on computer communication network applications and technologies, architectures, protocols, and algorithms. SIGCOMM '95 seeks papers about significant contributions to the broad field of computer and data communication networks. Authors are invited to submit full papers concerned with both theory and practice. Papers specifically focused on "higher-layer" issues of network infrastructure, management, and distributed application services are particularly encouraged. The areas of interest include, but are not limited to:

Topics

- Distributed application infrastructure paradigms
- Distributed common application services, middleware protocols
- Resource sharing, quality of service, multi-media networks
- Heterogeneous interworking, large scale networks
- Network management
- Important experimental results from operational networks
- High-speed networks, routing and addressing
- Wireless networking, support for mobile hosts
- Analysis/design of computer network architectures/algorithms
- Protocol specification, verification, and analysis

Submissions

SIGCOMM is a single-track, highly selective conference where successful submissions typically report results firmly substantiated by experiment, implementation, simulation, or mathematical analysis. The program committee is planning an excellent technical program and related activities. In addition to the presentation of papers and results, SIGCOMM '95 will offer tutorials and workshops by noted instructors on the two days preceding the actual conference. We also plan an evening session where speculative results and outrageous opinions can be presented and discussed. Papers must be less than 20 double-spaced pages (formatted for printing in the proceedings, papers may not be longer than 12 pages), have an abstract of 100–150 words, and be original material that has not been previously published nor is currently under review by another conference or journal. All submitted papers will be judged based on their quality and relevance through double-blind reviewing where the identities of the authors are withheld from the reviewers. Authors names should not appear on the paper or in the *PostScript* file for electronic submissions. A cover letter is required that identifies the paper title and lists the name, affiliation, telephone/fax numbers, and e-mail address of all authors. Authors of accepted papers need to sign an ACM copyright release form. The Proceedings of the conference will be published as a special issue of ACM SIGCOMM *Computer Communication Review*. The program committee may also select a few papers for possible publication in the IEEE/ACM *Transactions on Networking*. Five copies are required for paper submissions. Electronic submissions (preferred) should be uuencoded, compressed *PostScript*. Authors should separately e-mail the title, author names and abstract of their paper to the program chairs and identify any special equipment that will be required during its presentation. Due to the high number of anticipated submissions, authors are encouraged to strictly adhere to the submission date.

Paper submissions should be sent to:

David Clark and Karen Sollins, Program Co-Chairs
M.I.T. Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
U.S.A.
David Clark: +1 617 253 6003
Karen Sollins: +1 617 253 6006
Fax: +1 617 253 2673
E-mail: sc95pc@mercury.lcs.mit.edu

Tutorials

SIGCOMM '95 will begin with two days of tutorials/workshops, each of which is intended to cover a single topic in detail. Proposals are solicited from individuals willing to give tutorials, which may be either a half day (4 hours) or a full day in length and cover topics at an introductory or advanced level. Tutorial and workshop submissions should be made to the Tutorial Chair noted below and include an extended abstract and outline (2–4 pages), and an indication of length, objectives, and intended audience.

Student Paper Award

Papers submitted by students will enter a student-paper award contest. Among the accepted papers, a maximum of four outstanding papers will be awarded full conference registration and a travel grant of $500 US dollars. To be eligible the student must be the sole author of the paper, or the first author and primary contributor. A cover letter must identify the paper as a candidate for this competition.

Important dates

Paper submissions:	30 January 1995
Tutorial/workshop proposals:	30 January 1995
Notification of acceptance:	17 April 1995
Camera ready papers due:	22 May 1995

Write to *ConneXions*!

We'd love to hear your comments, suggestions and questions about anything you read in *ConneXions*. Our editorial address is given below. Use it for letters to the Editor, questions about back issues etc.:

ConneXions—The Interoperability Report
303 Vintage Park Drive, Suite 201
Foster City, CA 94404–1138, USA
Phone: +1 415-578-6900 or 1-800-INTEROP (Toll-free in the USA)
Fax: +1 415-525-0194
E-mail: connexions@interop.com
URL: http://www.interop.com

For questions about your subscription please call our customer service hotline: 1-800-575-5717 or +1 502-493-3217 outside the USA. This is the number for our subscription agency, the Cobb Group. Their fax number is +1 502-491-8050. The mailing address for subscription payments is P.O. Box 35840, Louisville, KY 40232–9496.

CONNEXIONS
303 Vintage Park Drive
Suite 201
Foster City, CA 94404-1138
Phone: 415-578-6900
FAX: 415-525-0194

ADDRESS CORRECTION
REQUESTED

FIRST CLASS MAIL
U.S. POSTAGE
PAID
SAN JOSE, CA
PERMIT NO. 1

CONNEXIONS

EDITOR and PUBLISHER Ole J. Jacobsen